NENA Standard for the Conveyance of Emergency Incident Data Objects (EIDOs) between Next Generation (NG9-1-1) Systems and Applications

Abstract: Standard defining how EIDOs are conveyed between Functional Elements

NENA Standard for the Conveyance of Emergency Incident Data Objects (EIDOs) between Next Generation (NG9-1-1) Systems and Applications

NENA-STA-024.1a-2023
DSC Approval: September 20, 2022
PRC Approval: January 10, 2023
NENA Board of Directors Approval: January 15, 2023
ANSI Approved: February 21, 2023
Next Scheduled Review Date: January 15, 2026

1 Executive Overview

The Emergency Incident Data Object (EIDO) is the JavaScript Object Notation (JSON) object that contains incident and related information intended to be passed between Functional Elements (FEs) for NENA-STA-021.1’s [16] stated purpose of processing emergency incidents, such as for use with systems handling 9-1-1 calls. EIDO itself constitutes a comprehensive mechanism for passing needed critical information fields between FEs and subsequently incident processing stakeholders. Without an equally reliable method of transport, EIDOs would be left stranded inside proprietary boxes or on disparate islands.

It is the purpose of the Standard for the Conveyance of Emergency Incident Data Objects to define a transport model for the secure, verified, and reliable delivery of EIDOs, both during the processing of emergency calls and thereafter. This version of EIDO Conveyance provides for a model of subscribers and notifiers that supports optional settings and filters used to set flow characteristics or limit notifications associated with incidents in which the subscriber is interested. EIDO Conveyance also provides a mechanism for on-demand requests for incident data. A future version of this standard may consider unsolicited notifications, e.g., push notifications.

Users involved in the exchange of incident information that use the EIDO to accomplish that exchange may be interested in this document’s described mechanisms for conveyance of EIDOs. EIDO Conveyance provides a crucial transport mechanism that offers the needed solution for passing an EIDO to authorized subscribers based on their incident scope or needs. For example, two FEs such as Incident Record Handling FEs and an Incident Data eXchange (IDX) would use subscriptions to share EIDOs on behalf of the users working on incidents related to calls handled by them. By defining a secure and reliable transport model for FEs to send EIDOs, EIDO Conveyance solves the problem of how to send the needed information to FEs that may not utilize the same vendor or function.

The reader should be familiar with general concepts of HTTP, JSON, and WebSocket to effectively understand this document and referenced concepts.
Table of Contents

1 EXECUTIVE OVERVIEW ............................................................................................................. 2

DOCUMENT TERMINOLOGY .......................................................................................................... 6

INTELLECTUAL PROPERTY RIGHTS (IPR) POLICY ................................................................. 7

REASON FOR ISSUE/REISSUE ..................................................................................................... 7

2 EMERGENCY INCIDENT DATA OBJECT CONVEYANCE ......................................................... 10

2.1 WebSocket-based Incident Data Subscription ...................................................................... 10

2.1.1 General Description ........................................................................................................... 10

2.1.2 URI scheme ...................................................................................................................... 11

2.1.3 Client WebSocket Handshake ............................................................................................ 12

2.1.4 Client WebSocket API Actions ........................................................................................ 13

2.1.5 Objects ............................................................................................................................. 14

2.1.6 Notification Model ........................................................................................................... 22

2.1.7 Use Cases ........................................................................................................................ 25

2.2 MinRates and MaxRates in Detail ....................................................................................... 27

2.2.1 minRate ............................................................................................................................ 27

2.2.2 maxRate ........................................................................................................................... 28

2.2.3 Subscription maxRate Example ....................................................................................... 30

2.3 QUALFilter ........................................................................................................................... 32

2.3.1 Applying a qualFilter ....................................................................................................... 34

2.3.2 Example ............................................................................................................................ 35

2.4 Transport in Call Signaling .................................................................................................. 37

2.5 EIDO Dereference Factory .................................................................................................. 38

2.6 EIDO Dereference Service ................................................................................................... 39

2.6.1 Request and Response ..................................................................................................... 39

2.7 EIDO Retrieval Service ......................................................................................................... 39

2.8 Data Rights Management ..................................................................................................... 40

2.9 Logging ................................................................................................................................ 41

2.9.1 EidoLogEvent .................................................................................................................... 42

2.9.2 EidoDereferenceFactoryQueryLogEvent ....................................................................... 42

2.9.3 EidoDereferenceFactoryQueryResponseLogEvent ......................................................... 42

2.9.4 EidoDeniedLogEvent ........................................................................................................ 42

2.9.5 EidoTransmissionErrorLogEvent .................................................................................... 43

2.9.6 SubscriptionRequestedLogEvent ....................................................................................... 43

2.9.7 SubscriptionRequestedResponseLogEvent ...................................................................... 43

2.9.8 SubscriptionRequestedLogEvent ....................................................................................... 43

2.9.9 SubscriptionTerminatedResponseLogEvent ................................................................. 44

2.9.10 WebSocketEstablishedLogEvent ..................................................................................... 44

2.9.11 WebSocketTerminatedLogEvent ..................................................................................... 44

2.10 Security .................................................................................................................................. 44

3 IANA CONSIDERATIONS ........................................................................................................... 44

© 2023 National Emergency Number Association, Inc.
3.1 EVENT PACKAGE REGISTRY ................................................................. 44
3.2 INTERFACE NAMES REGISTRY .......................................................... 45
3.3 EIDO DEREFS D RATION DENIAL REASON REGISTRY .................. 45
3.4 EIDO TRANSMISSION FAILURE REGISTRY ..................................... 45
3.5 WebSocket Subprotocol Name Registry ............................................. 45

4 IMPACTS, CONSIDERATIONS, ABBREVIATIONS, TERMS, AND DEFINITIONS .......... 46
4.1 OPERATIONS IMPACTS SUMMARY .................................................... 46
4.2 TECHNICAL IMPACTS SUMMARY ...................................................... 46
4.3 SECURITY IMPACTS SUMMARY .......................................................... 46
4.4 RECOMMENDATION FOR ADDITIONAL DEVELOPMENT WORK .......... 46
  4.4.1 Unsolicited EIDO Conveyance ..................................................... 46
  4.4.2 Deltas ......................................................................................... 47
  4.4.3 Location and Content Filters ....................................................... 47
  4.4.4 qualFilter Examples ................................................................ 47
  4.4.5 Signatures .................................................................................. 47
  4.4.6 Discrepancy Reporting ............................................................... 47
  4.4.7 Subscribe to Resource ................................................................. 47
  4.4.8 Removal of EIDO Dereference Factory and EIDO Dereference Service .... 47
4.5 ANTICIPATED TIMELINE ................................................................. 47
4.6 COST FACTORS .............................................................................. 47
4.7 COST RECOVERY CONSIDERATIONS ............................................ 48
4.8 ADDITIONAL IMPACTS (NON-COST RELATED) ............................... 48
4.9 ABBREVIATIONS, TERMS, AND DEFINITIONS ................................. 48

5 REFERENCES ...................................................................................... 50

6 EXHIBIT ............................................................................................. 51

7 APPENDIX A: SUBSCRIPTION FILTER EXAMPLE .............................................. 52

ACKNOWLEDGMENTS ............................................................................. 60

© 2023 National Emergency Number Association, Inc.
This Standard Document (STA) is published by the National Emergency Number Association (NENA) as an information source for 9-1-1 System Service Providers, network interface vendors, system vendors, telecommunication service providers, and 9-1-1 Authorities. As an industry Standard it provides for interoperability among systems and services adopting and conforming to its specifications.

NENA reserves the right to revise this Standard Document for any reason including, but not limited to:

- Conformity with criteria or standards promulgated by various agencies,
- Utilization of advances in the state of the technical arts,
- Reflecting changes in the design of equipment, network interfaces, or services described herein.

This document is an information source for the voluntary use of communication centers. It is not intended to be a complete operational directive.

It is possible that certain advances in technology or changes in governmental regulations will precede these revisions. All NENA documents are subject to change as technology or other influencing factors change. Therefore, this NENA document should not be the only source of information used. NENA recommends that readers contact their 9-1-1 System Service Provider (9-1-1 SSP) representative to ensure compatibility with the 9-1-1 network, and their legal counsel, to ensure compliance with current regulations.

Patents may cover the specifications, techniques, or network interface/system characteristics disclosed herein. No license is granted, whether expressed or implied. This document shall not be construed as a suggestion to any manufacturer to modify or change any of its products, nor does this document represent any commitment by NENA, or any affiliate thereof, to purchase any product, whether or not it provides the described characteristics.

By using this document, the user agrees that NENA will have no liability for any consequential, incidental, special, or punitive damages arising from use of the document.

NENA’s Committees have developed this document. Recommendations for changes to this document may be submitted to:

National Emergency Number Association
1700 Diagonal Rd, Suite 500
Alexandria, VA 22314
202.466.4911
or commleadership@nena.org
Document Terminology

This section defines keywords, as they should be interpreted in NENA documents. The form of emphasis (UPPER CASE) shall be consistent and exclusive throughout the document. Any of these words used in lower case and not emphasized do not have special significance beyond normal usage.

1. MUST, SHALL, REQUIRED: These terms mean that the definition is a normative (absolute) requirement of the specification.

2. MUST NOT: This phrase, or the phrase "SHALL NOT", means that the definition is an absolute prohibition of the specification.

3. SHOULD: This word, or the adjective "RECOMMENDED", means that there may exist valid reasons in particular circumstances to ignore a particular item, but the full implications must be understood and carefully weighed before choosing a different course.

4. SHOULD NOT: This phrase, or the phrase "NOT RECOMMENDED" means that there may exist valid reasons in particular circumstances when the particular behavior is acceptable or even useful, but the full implications should be understood and the case carefully weighed before implementing any behavior described with this label.

5. MAY: This word, or the adjective "OPTIONAL", means that an item is truly optional. One vendor may choose to include the item because a particular marketplace requires it or because the vendor feels that it enhances the product while another vendor may omit the same item. An implementation which does not include a particular option “must” be prepared to interoperate with another implementation which does include the option, though perhaps with reduced functionality. In the same vein an implementation which does include a particular option “must” be prepared to interoperate with another implementation which does not include the option (except, of course, for the feature the option provides.)

These definitions are based on RFC 2119 [6].
Intellectual Property Rights (IPR) Policy

NOTE – The user’s attention is called to the possibility that compliance with this standard may require use of an invention covered by patent rights. By publication of this standard, NENA takes no position with respect to the validity of any such claim(s) or of any patent rights in connection therewith. If a patent holder has filed a statement of willingness to grant a license under these rights on reasonable and nondiscriminatory terms and conditions to applicants desiring to obtain such a license, then details may be obtained from NENA by contacting the Committee Resource Manager identified on NENA’s website at www.nena.org/ipr.

Consistent with the NENA IPR and Antitrust Policy, available at www.nena.org/ipr, NENA invites any interested party to bring to its attention any copyrights, patents or patent applications, or other proprietary rights that may cover technology that may be required to implement this standard.

Please address the information to:

National Emergency Number Association  
1700 Diagonal Rd, Suite 500  
Alexandria, VA 22314  
202.466.4911  
or commleadership@nena.org

Reason for Issue/Reissue

NENA reserves the right to modify this document. Upon revision, the reason(s) will be provided in the table below.

<table>
<thead>
<tr>
<th>Document Number</th>
<th>Approval Date</th>
<th>Reason for Issue/Reissue</th>
</tr>
</thead>
<tbody>
<tr>
<td>NENA-STA-024.1-2023</td>
<td>01/15/2023</td>
<td>Initial Document</td>
</tr>
</tbody>
</table>
Publishing error corrections:

1) Syntax: wss://<target host>/ent<eido subscribe path>

2) Example:
   wss://idx.esinet.net/IncidentData/ents subscribe

3) The response to the request, logged via a
   EidoDereferenceFactoryQueryResponseLogEvent, MUST include the
   “direction” member, the “queryId” member, and either a “generatedURI”
   member or, a “responseError” member that contains an error code
   from the Error Codes Registry and a
   “responseText” in the case of an error.

4) an “incidentId” member in the header field of the log record, and a
   “reasonCode” member containing an associated reason code from the EIDO
   Dereference Denial registry and a
   “reasonText” member.

5) an “incidentId” member in the header field of the log record, and a
   “errorCode status” member containing a status code from the EIDO
   Transmission Status registry indicating the resultant failure and an
   “errorText” member.

6) If the subscribe request was denied
   the log event MUST include an error message in the
   “errorCode_responseError” and
   “errorText” members respectively.
<table>
<thead>
<tr>
<th>Document Number</th>
<th>Approval Date</th>
<th>Reason for Issue/Reissue</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>7) a &quot;statusCode&quot; member and a &quot;statusTextDescription&quot; member.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>8) and an optional “closeTextReason” member containing a reason per RFC 6455 [12].</td>
</tr>
<tr>
<td></td>
<td></td>
<td>9) See NENA Master Glossary of 9-1-1 Terminology, NENA-ADM-000 Error! Reference source not found. Error! Reference source not found, for a complete listing of terms used in NENA documents.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>10) <a href="https://idx.esinet.net/IncidentDate/v1/incidentpsap.alleghany.pa.us/echo/uuid/urn:emergency:uid:incident:6955@psap.alleghany.pa.us">https://idx.esinet.net/IncidentDate/v1/incidentpsap.alleghany.pa.us/echo/uuid/urn:emergency:uid:incident:6955@psap.alleghany.pa.us</a></td>
</tr>
<tr>
<td></td>
<td></td>
<td>11) Host: psap.alleghany.pa.us</td>
</tr>
</tbody>
</table>
2 Emergency Incident Data Object Conveyance

PSAPs will make extensive use of NENA-STA-021 NG9-1-1 Emergency Incident Data Object (EIDO) standard [16]. Most interactions between FEs inside a PSAP, between PSAPs, and between PSAPs and other entities, involve sending and receiving an EIDO.

All transmissions of EIDOs MUST use TLS security as per NENA-STA-010 [16].

The content of an EIDO MUST conform to the data rights management policy of the owner of the data per the Authorization and Data Rights Management section of NENA-STA-010 [16].

For interoperability, all implementations SHALL accept EIDOs of any size up to and including 10,485,760 bytes.

2.1 WebSocket-based Incident Data Subscription

A Functional Element (FE) exposes an interface for subscribing to events or messages as matched by the provided filter [25]. Using the qualFilter the service will support a subscription to matching specific target objects that will be sent via a WebSocket.

2.1.1 General Description

Clients establish a connection to a service via a WebSocket per RFC 6455 [12] specifying a WebSocket Protocol named “emergency-ent”. WebSocket is used in order to provide an asynchronous, full duplex communication path between functional elements, as opposed to a unidirectional, synchronous protocol such as HTTP.

Mutual authentication with credentials traceable to the PSAP Credentialing Agency (PCA) is required for all connections.

Once the WebSocket has been established the client may request one or more subscriptions to incident data based on one of the available methods: a subscription that returns one EIDO for all new incidents or a filtered subset thereof (New Incidents), or a subscription for all EIDOs for a specific incident or a rate-limited subset thereof (Single Incident).

For interoperability, all implementations SHALL accept subscription requests of any size up to and including 65,536 bytes.

All communication related to a subscription is handled over the WebSocket.

Upon successful subscription, notifications SHALL be sent to the subscriber as they match the criteria associated with the subscription until the subscription expires or an unsubscribe
occurs. For a new incidents subscription, a notification SHALL be sent immediately that contains all active incidents that match the criteria. If no incidents match the criteria, an empty notification SHALL be sent as described in the minRate section below. For a Single Incident subscription, a notification SHALL be sent immediately that contains that incident. A Single Incident subscription SHALL be accepted for a closed incident if the request is made within five minutes of its closure, in which case a single EIDO representing the last state of that incident SHALL be sent and no further notifications will be made for that incident; however, a notifier MAY accept a Single Incident subscription for an incident that has been closed longer than five minutes.

A notifier will not send any notifications until a successful subscription has taken place. Notifications will only be sent if they match the filter or other subscription criteria.

A subscriber SHALL send subsequent subscribe actions to reset the expiry. To reset the expiry a request MUST contain the subscription identifier provided when originally subscribing. The new expiration is provided back in the response. An empty notification is used as a keep-alive mechanism for long term subscriptions.

Any change to a subscription by a subscriber other than resetting the expiry requires creating a new subscription and then sending an unsubscribe message for the old subscription.

Once established, a WebSocket SHALL handle multiple subscriptions.

Upon expiration of the last subscription still in effect the notifier will close the WebSocket. It is up to the subscriber to detect this closure and act appropriately.

If the WebSocket is closed (including if the underlying TCP socket is closed) then the notifier shall flush all subscriptions established using that WebSocket without notifying the subscriber and the subscriber shall reopen the WebSocket and reestablish any subscriptions still desired as new subscriptions.

2.1.2 URI scheme

A URI SHALL exist for each Functional Element that is an EIDO notifier.

The external subscription URI for an agency or service is discoverable using the service/agency locator as specified in STA-010 [16].

The Secure WebSocket URI Scheme is used for subscriptions per RFC 3986 [7].

Multiple notifier functional elements can share the same subscription interface, the implementation of which is outside the scope of this standard.

Syntax: wss://<target host>/ent

Example: wss://idx.esinet.net/IncidentData/ent
2.1.3 Client WebSocket Handshake

2.1.3.1 Client WebSocket Negotiation

2.1.3.1.1 Negotiate WebSocket

Open a WebSocket to a service endpoint.

2.1.3.1.2 HTTP Headers

The following HTTP headers are REQUIRED to establish a WebSocket for the WebSocket Protocol “emergency-ent” per RFC 6455, *The WebSocket Protocol* [12]:

<table>
<thead>
<tr>
<th>Header Name</th>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Connection</td>
<td>“Upgrade”</td>
<td>Directive to change protocol.</td>
</tr>
<tr>
<td>Upgrade</td>
<td>“websocket”</td>
<td>Directive to upgrade to the WebSocket protocol.</td>
</tr>
<tr>
<td>Host</td>
<td>Hostname</td>
<td>Identification of the host requested.</td>
</tr>
<tr>
<td>Sec-WebSocket-Protocol</td>
<td>“emergency-ent1.0”</td>
<td>Used to identify the version(s) of the message protocol supported by the requestor.</td>
</tr>
<tr>
<td>Sec-WebSocket-Key</td>
<td>base64</td>
<td>The value of this header field MUST be a nonce consisting of a randomly selected 16-byte value that has been base64-encoded (see Section 4 of [8][8]. The nonce MUST be selected randomly for each connection.</td>
</tr>
<tr>
<td>Sec-WebSocket-Version</td>
<td>“13”</td>
<td>WebSocket protocol version number.</td>
</tr>
</tbody>
</table>

The following HTTP headers will be returned in response to the request to establish a WebSocket:

<table>
<thead>
<tr>
<th>Header Name</th>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Connection</td>
<td>“Upgrade”</td>
<td>The server is upgrading the connection.</td>
</tr>
</tbody>
</table>
### Header Name | Value | Description
--- | --- | ---
Upgrade | "websocket" | The server is upgrading the connection to a WebSocket.
Set-WebSocket-Accept | Base64 | Sent from the server to the client to confirm that the server is willing to initiate the WebSocket connection.
Sec-WebSocket-Protocol | "emergency-ent1.0" | Used to identify the message protocol and its version.

**Example: WebSocket Establishment Request**

```plaintext
GET /eido_urn HTTP/1.1
Host: idx.esinet.net
Connection: Upgrade
Upgrade: websocket
Sec-WebSocket-Protocol: emergency-ent1.0
Sec-WebSocket-Key: dGhlIHNhbXBsZSBub25jZQ==
Sec-WebSocket-Version: 13
```

**Example: WebSocket Establishment Response**

```
HTTP/1.1 101 Switching Protocols
Upgrade: websocket
Connection: Upgrade
Sec-WebSocket-Accept: s3pPLMBiTxaQ9kYGzzhZRbK+xOo=
Sec-WebSocket-Protocol: emergency-ent1.0
```

### 2.1.4 Client WebSocket API Actions

Following the establishment of the WebSocket, the client sends messages to the FE using WebSocket text frames, which are encoded using UTF-8. The methods defined herein are subscribe and unsubscribe.

<table>
<thead>
<tr>
<th>Payload Object</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>subscribe:{&lt;subscription_object&gt;}</td>
<td>Apply an EIDO subscription for specific results. A subscription ID is provided back in an Acknowledgement and used to keep a subscription alive or make subsequent filter changes.</td>
</tr>
<tr>
<td>unsubscribe:{&lt;subscriptionId&gt;}</td>
<td>Unsubscribe from the given subscription ID.</td>
</tr>
</tbody>
</table>
If a request is made for any method other than subscribe or unsubscribe then that notifier shall respond with the following:

```json
{
  "response": {
    "requestId": "63bfd612-f558-4489-b653-caa76a7ed9b",
    "statusCode": 501,
    "statusText": "Not implemented"
  }
}
```

If no requestId is specified in the request, then no requestId element will be included in the response.

### 2.1.5 Objects

There are eight types of conveyance objects. They include:

1. event
2. eventResponse
3. subscribe
4. subscribeResponse
5. terminate
6. terminateResponse
7. unsubscribe
8. unsubscribeResponse

The followings sections provide more detail to describe the properties present in each of these objects.

#### 2.1.5.1 subscribe

##### 2.1.5.1.1 client request

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Condition</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>requestId</td>
<td>string</td>
<td>MANDATORY</td>
<td>An ID generated by the subscriber that will be returned by the notifier in its response to the request.</td>
</tr>
<tr>
<td>Name</td>
<td>Type</td>
<td>Condition</td>
<td>Description</td>
</tr>
<tr>
<td>--------------</td>
<td>-------</td>
<td>----------------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>requestType</td>
<td>string</td>
<td>CONDITIONAL REQUIRED for a new subscription (i.e., if no subscriptionId provided), otherwise, MUST be omitted.</td>
<td>Type of request, e.g., &quot;EIDO&quot;.</td>
</tr>
<tr>
<td>requestSubType</td>
<td>string</td>
<td>CONDITIONAL</td>
<td>If requestType is omitted, then requestSubType MUST be omitted. If requestType is &quot;EIDO&quot;, requestSubType MUST be present and MUST be one of “single” or “new”. Other request types may have their own conditions or may not require a subtype.</td>
</tr>
<tr>
<td>requestAccepts</td>
<td>string</td>
<td>CONDITIONAL REQUIRED for a new subscription, i.e., if subscriptionId is not specified.</td>
<td>The MIME type associated with the requestType and the major versions thereof accepted in the notification payload as a quoted comma-delimited list, as in the Accept HTTP header per RFC 7231 [13], e.g., application/emergency.eido+json; version=“1” or application/emergency.eido+json; version=“1,2,3”. The version is required and is the only extension parameter permitted.</td>
</tr>
<tr>
<td>Name</td>
<td>Type</td>
<td>Condition</td>
<td>Description</td>
</tr>
<tr>
<td>--------------------</td>
<td>---------</td>
<td>-------------------------</td>
<td>-----------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>qualFilter</td>
<td>string</td>
<td>CONDITIONAL</td>
<td>A qualFilter MUST NOT be specified if an incidentId is specified. If requestType is “EIDO” and incidentId is not specified.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>OPTIONAL</td>
<td>When subscribing for new incidents, the qualFilter may be used to be notified of a subset thereof.</td>
</tr>
<tr>
<td>qualFilterEvalPeriod</td>
<td>integer</td>
<td>CONDITIONAL OPTIONAL if qualFilter specified, otherwise MUST NOT be specified.</td>
<td>If qualFilter specified, a positive integer specifying the requested number of seconds between qualFilter evaluations.</td>
</tr>
<tr>
<td>minRate</td>
<td>integer</td>
<td>CONDITIONAL OPTIONAL if and only if a subscriptionId is not specified, otherwise MUST NOT be specified.</td>
<td>Number of seconds after the last notification after which a notification must be sent consistent with RFC 6446 [10]. If not specified there is no minimum rate.</td>
</tr>
<tr>
<td>maxRate</td>
<td>integer</td>
<td>CONDITIONAL OPTIONAL if and only if a subscriptionId is not specified, otherwise MUST NOT be specified.</td>
<td>Minimum time in seconds to wait between notifications which is the maximum rate at which notifications are sent consistent with RFC 6446 [10]. If not specified there is no maximum rate.</td>
</tr>
</tbody>
</table>
### Name | Type | Condition | Description
--- | --- | --- | ---
subscriptionId | string | OPTIONAL | The subscriptionId previously returned by the notifier in the subscribeResponse. This is used for keeping a subscription alive.
incidentId | string | CONDITIONAL. For ONLY request type “EIDO” and subtype “single”, MUST be present to subscribe to a specific incident. MUST not be present for other subtypes | The Incident Tracking Identifier associated with the call.
expires | integer | OPTIONAL | Number of seconds after which the subscription shall expire.

**client subscription request filter example:**

```json
{
    "subscribe":{
        "requestId":"63bfd612-f558-4489-b653-caa76a7ed9b",
        "requestType":"EIDO",
        "requestSubType":"new",
        "requestAccepts":"application/emergency.eido+json;version="1,2,3",
        "qualFilter":"
        <filter_string>
        
        "minRate":1500
    }
}
```

**client reset expire timer example:**

```json
{
    "subscribe":{
        "requestId":"63bfd612-f558-4489-b653-caa76a7ed9b",
        "subscriptionId":"sub1234",
    }
}
```
client subscription request for an incidentId example:

```json
{
    "subscribe": {
        "requestId": "63bfd612-f558-4489-b653-caa76a7edb9b",
        "requestType": "EIDO",
        "requestSubType": "single",
        "requestAccepts": "application/emergency.eido+json; version="1,2,3",
        "incidentId": "urn:emergency:uid:incidentid:a56e556d871:bcf.state.pa.us"
    }
}
```

**Note** – The request may require an identifier related to a discrete object. For EIDO, incidentId is used to subscribe to a specific incident ID. For any other transport use case it is an event level identifier as defined by that specific protocol standard, e.g., Service or Element State Event notification it can be the SIP Call-ID of the SUBSCRIBE/NOTIFY dialog.

### 2.1.5.1.2 Subscribe Response - “subscribe” service response

#### 2.1.5.1.2.1 subscribeResponse

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Condition</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>requestId</td>
<td>string</td>
<td>MANDATORY</td>
<td>The ID provided by the subscriber.</td>
</tr>
<tr>
<td>subscriptionId</td>
<td>string</td>
<td>CONDITIONAL, REQUIRED if subscription is accepted</td>
<td>The globally unique identifier for the subscription.</td>
</tr>
<tr>
<td>qualFilterEvalPeriod</td>
<td>integer</td>
<td>CONDITIONAL, REQUIRED if qualFilter specified</td>
<td>Negotiated number of seconds between qualFilter evaluations. If requested, may be a longer period than the request but not shorter.</td>
</tr>
<tr>
<td>minRate</td>
<td>integer</td>
<td>CONDITIONAL, REQUIRED if specified by the subscriber</td>
<td>May be a longer period than requested but not shorter.</td>
</tr>
</tbody>
</table>
The response to the subscription request SHALL contain one of the following status codes.

<table>
<thead>
<tr>
<th>Status Code</th>
<th>Description</th>
<th>Subscriber Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>200</td>
<td>OK</td>
<td>None</td>
</tr>
<tr>
<td>400</td>
<td>Bad Request</td>
<td>None</td>
</tr>
<tr>
<td>403</td>
<td>Forbidden</td>
<td>None</td>
</tr>
<tr>
<td>404</td>
<td>Not Found</td>
<td>None</td>
</tr>
<tr>
<td>406</td>
<td>Unacceptable</td>
<td>None</td>
</tr>
<tr>
<td>413</td>
<td>Payload too large</td>
<td>None</td>
</tr>
<tr>
<td>481</td>
<td>Subscription does not exist</td>
<td>None</td>
</tr>
<tr>
<td>500</td>
<td>Internal Server Error</td>
<td>None</td>
</tr>
<tr>
<td>503</td>
<td>Service Unavailable</td>
<td>Retry request</td>
</tr>
</tbody>
</table>

Successful “subscribe” service response example
Unsuccessful “subscribe” service response example

```
{
    "subscribeResponse": {
        "requestId": "63bfd612-f558-4489-b653-caa76a7edb9b",
        "subscriptionId": "sub1234",
        "expires": 3600,
        "statusCode": 481,
        "statusText": "Subscription does not exist"
    }
}
```

2.1.5.2 unsubscribe

2.1.5.2.1 client request

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Condition</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>requestId</td>
<td>string</td>
<td>mandatory</td>
<td>An ID generated by the subscriber that will be returned by the notifier in its response to the request.</td>
</tr>
<tr>
<td>subscriptionId</td>
<td>string</td>
<td>mandatory</td>
<td>The subscriptionId sent by the service. This is provided back in an Acknowledgement by the service, used for keeping a subscription alive or subsequent filter modifications.</td>
</tr>
</tbody>
</table>

Client unsubscribe example:

```
{
    "unsubscribe": {
        "requestId": "63bfd612-f558-4489-b653-caa76a7edb9b",
        "subscriptionId": "sub1234"
    }
}
```
2.1.5.2.2 “unsubscribe” service response

2.1.5.2.2.1 unsubscribeResponse

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Condition</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>requestId</td>
<td>string</td>
<td>MANDATORY</td>
<td>The ID specified by the subscriber.</td>
</tr>
<tr>
<td>subscriptionId</td>
<td>string</td>
<td>MANDATORY</td>
<td>Globally unique identifier for the subscription.</td>
</tr>
<tr>
<td>statusCode</td>
<td>integer</td>
<td>MANDATORY</td>
<td>Status code of response according to the request, e.g., 200, as defined in</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>the Hypertext Transfer Protocol Status Code Registry (RFC 7231) [13] or the</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>NENA Registry System StatusCodes Registry [14].</td>
</tr>
<tr>
<td>statusText</td>
<td>string</td>
<td>MANDATORY</td>
<td>The statusText is intended to provide information related to the statusCode.</td>
</tr>
</tbody>
</table>

The response to the unsubscribe request SHALL contain one of the following status codes.

<table>
<thead>
<tr>
<th>Status Code</th>
<th>Description</th>
<th>Subscriber Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>200</td>
<td>OK</td>
<td>None</td>
</tr>
<tr>
<td>400</td>
<td>Bad Request</td>
<td>None</td>
</tr>
<tr>
<td>413</td>
<td>Payload too large</td>
<td>None</td>
</tr>
<tr>
<td>481</td>
<td>Subscription does not exist</td>
<td>None</td>
</tr>
<tr>
<td>500</td>
<td>Internal Server Error</td>
<td>None</td>
</tr>
<tr>
<td>503</td>
<td>Service Unavailable</td>
<td>Retry request</td>
</tr>
</tbody>
</table>

Successful “unsubscribe” service response:

```json
{
    "unsubscribeResponse":{
        "subscriptionId": "sub1234",
        "statusCode": 200,
        "statusText": "Unsubscribed"
    }
}
```

Unsuccessful “unsubscribe” service response:
2.1.6 Notification Model

Based on the subscription, qualifying incidents will be sent to the subscriber in the form of a notification that contains an event. The subscriber will respond with an eventResponse. A notifier sends a terminate message when a subscription is terminated such as due to subscription expiration or when an incident is closed by the notifier. An explicit unsubscribe request by the subscriber will not cause the notifier to generate a terminate message.

### 2.1.6.1 Notification Objects

<table>
<thead>
<tr>
<th>Payload Object</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>event:{&lt;event_object&gt;}</td>
<td>Notification has been received.</td>
</tr>
<tr>
<td>terminate:{&lt;subscriptionId&gt;}</td>
<td>Notification of termination of a subscription.</td>
</tr>
</tbody>
</table>

### 2.1.6.2 Event Object

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>subscriptionId</td>
<td>string</td>
<td>Subscription to which this event belongs.</td>
</tr>
<tr>
<td>transactionId</td>
<td>string</td>
<td>A GUID used to identify a notification for acknowledgement.</td>
</tr>
<tr>
<td>notification</td>
<td>array</td>
<td>Array of objects, e.g., EIDOs.</td>
</tr>
</tbody>
</table>

An example of a notification is:

```json
{
   "event": {
      "subscriptionId": "sub1234",
      "transactionId": "f894fjhdsh940jdf@192.168.2.1",
      "notification": [
         <EIDO>
      ]
   }
}
```
2.1.6.3 Event Notification Response

2.1.6.3.1 Event Response

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>transactionId</td>
<td>string</td>
<td>GUID used to identify a transaction for acknowledgement.</td>
</tr>
<tr>
<td>statusCode</td>
<td>integer</td>
<td>Status code of response according to the request, e.g., 200, as defined in the Hypertext Transfer Protocol Status Code Registry (RFC 7231) [13] or the NENA Registry System StatusCodes Registry [14].</td>
</tr>
<tr>
<td>statusText</td>
<td>string</td>
<td>The statusText is intended to provide information related to the statusCode.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Status Code</th>
<th>Description</th>
<th>Notifier Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>200</td>
<td>OK</td>
<td>None</td>
</tr>
<tr>
<td>400</td>
<td>Bad Request</td>
<td>None</td>
</tr>
<tr>
<td>404</td>
<td>Not Found</td>
<td>None</td>
</tr>
<tr>
<td>413</td>
<td>Payload too large</td>
<td>Send an event for each notification object without regard to rate limiting</td>
</tr>
<tr>
<td>481</td>
<td>Subscription does not exist</td>
<td>Do not send notifications for that subscription</td>
</tr>
<tr>
<td>500</td>
<td>Internal Server error</td>
<td>None</td>
</tr>
<tr>
<td>501</td>
<td>Not implemented by subscriber</td>
<td>None</td>
</tr>
<tr>
<td>503</td>
<td>Service Unavailable</td>
<td>Retry</td>
</tr>
</tbody>
</table>

2.1.6.3.2 Subscriber Notification Response example:

```json
{
    "eventResponse": {
        "transactionId": "f894fjhdsh940jdf@192.168.2.1",
        "statusCode": 200,
        "statusText": "OK"
    }
}
```
2.1.6.4 Terminate Object

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>requestId</td>
<td>string</td>
<td>An ID generated by the notifier that will be returned by the subscriber in its response to the request.</td>
</tr>
<tr>
<td>subscriptionId</td>
<td>string</td>
<td>Subscription being terminated.</td>
</tr>
</tbody>
</table>

An example of a termination is:

```json
{
   "terminate": {
      "requestId": "63bfd612-f558-4489-b6253-caa76a7ed9b",
      "subscriptionId": "sub1234"
   }
}
```

2.1.6.5 Terminate Notification Response

2.1.6.5.1 Terminate Response

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>requestId</td>
<td>string</td>
<td>The ID generated by the notifier.</td>
</tr>
<tr>
<td>subscriptionId</td>
<td>string</td>
<td>Subscription being terminated.</td>
</tr>
<tr>
<td>statusCode</td>
<td>integer</td>
<td>Status code of response according to the request, e.g., 200, as defined in the Hypertext Transfer Protocol Status Code Registry (RFC 7231) [13] or the NENA Registry System StatusCodes Registry [14].</td>
</tr>
<tr>
<td>statusText</td>
<td>string</td>
<td>The statusText is intended to provide information related to the statusCode.</td>
</tr>
</tbody>
</table>

The response to the terminate request SHALL contain one of the following status codes.

<table>
<thead>
<tr>
<th>Status Code</th>
<th>Description</th>
<th>Notifier Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>200</td>
<td>OK</td>
<td>None</td>
</tr>
<tr>
<td>400</td>
<td>Bad Request</td>
<td>None</td>
</tr>
<tr>
<td>Status Code</td>
<td>Description</td>
<td>Notifier Action</td>
</tr>
<tr>
<td>-------------</td>
<td>--------------------------------------------</td>
<td>--------------------------------------</td>
</tr>
<tr>
<td>481</td>
<td>Subscription does not exist</td>
<td>Do not send notifications for that subscription</td>
</tr>
<tr>
<td>500</td>
<td>Internal Server error</td>
<td>None</td>
</tr>
<tr>
<td>501</td>
<td>Not implemented by subscriber</td>
<td>None</td>
</tr>
<tr>
<td>503</td>
<td>Service Unavailable</td>
<td>Retry</td>
</tr>
</tbody>
</table>

### 2.1.6.5.2 Terminate Response example:

```
{
  "terminateResponse": {
    "requestId": "63bfd612-f558-4489-b653-caa76a7ed9b",
    "subscriptionId": "sub1234",
    "statusCode": 200,
    "statusText": "OK"
  }
}
```

### 2.1.7 Use Cases

<table>
<thead>
<tr>
<th>Use Case</th>
<th>Response</th>
</tr>
</thead>
<tbody>
<tr>
<td>Notifier receives a subscribe message that does not contain a subscriptionId on a WebSocket.</td>
<td>The subscription is accepted by the notifier.</td>
</tr>
</tbody>
</table>
|                                                                         | {
|                                                                         |   "subscribeResponse": {
|                                                                         |     "requestId": "63bfd612-f558-4489-b653-caa76a7ed9b",
|                                                                         |     "subscriptionId": "sub1234",
|                                                                         |     "expires": 3600,
|                                                                         |     "qualFilterEvalPeriod": 10,
|                                                                         |     "minRate": 1500,
|                                                                         |     "statusCode": 200,
|                                                                         |     "statusMessage": "Subscription Accepted"
|                                                                         |   }
```
<table>
<thead>
<tr>
<th>Use Case</th>
<th>Response</th>
</tr>
</thead>
<tbody>
<tr>
<td>Notifier receives a subscribe message that contains a subscriptionId but the subscriptionId does not match an active subscriptionId on the established WebSocket.</td>
<td>The functional element returns a subscribeResponse message with a statusCode of 481 indicating that the subscription was rejected.</td>
</tr>
<tr>
<td>Functional element receives a subscribe message that contains a subscriptionId that matches the active subscriptionId. The filter contained within the subscribe message is different than the active filter.</td>
<td>The functional element returns a successful subscribe response message. Ex:</td>
</tr>
</tbody>
</table>

```json
{  
  "subscribe": {  
    "requestId": "63bfd612-f558-4489-b653-caa76a7edeb9b",  
    "subscriptionId": "sub4321",  
    "minRate": 1500  
  }  
}
```

The updated filter replaces the existing active filter.
Use Case

A functional element receives a subscribe message for an incident ID that is unknown.

```
{  
  "subscribe": {  
    "requestId": "63bfd612-f558-4489-b653-caa76a7edeb9b",  
    "requestType": "EIDO",  
    "requestSubType": "single",  
    "requestAccepts": "application/emergency.eido+json; version=\"1,2,3\"",  
    "incidentId": "urn:emergency:uid:incidentid:a56e556d871:bcf.state.pa.us"  
  }  
}
```

The functional element returns a subscribeResponse message with a statusCode of 404 indicating that the subscription was rejected because of the unknown incident.

```
{  
  "subscribeResponse": {  
    "requestId": "63bfd612-f558-4489-b653-caa76a7edeb9b",  
    "incidentId": "event1234",  
    "expires": 0,  
    "statusCode": 404,  
    "statusMessage": "Incident does not exist"  
  }  
}
```

2.2 MinRates and MaxRates in Detail

2.2.1 minRate

minRate refers to the minimum rate that notification messages must be sent. It is specified as a period in seconds. If a subscription notification has not been sent by the notifier within the minRate period, then the notifier must send an empty event notification message consisting of a message similar to the following example:

```
{  
  "event": {  
    "subscriptionId": "sub1234",  
    "transactionId": "f894fjhdsh940jdf@192.168.2.1",  
    "notification": []  
  }  
}
```

2.2.1.1 Subscriber Behavior

A subscriber that wishes to request a specific minimum notification rate MUST populate the minRate field in the subscribe message with a value greater than 0. The subscriber must be aware that the subscribe response message contains the actual minRate to be used for...
the subscription. The notifier can respond with a different rate than that requested but cannot make it a shorter period than the requested rate.

2.2.1.2 Notifier Behavior

A notifier MUST consider the minRate parameter specified by the subscriber and apply local policy to determine if the specified minRate is permitted. The notifier MAY decide to increase the proposed minRate period based on its local policy, static configuration, or other implementation-determined constraints. The notifier MUST reflect the effective minimum notification period in the minRate parameter of the subscribe response message.

A notifier MUST generate an empty notification message when the time since the last notify message exceeds the agreed upon minRate period.

2.2.2 maxRate

The maxRate refers to the maximum rate that notification messages can be sent. It is specified as a period in seconds between notifications. If the effective maxRate value is 0 then maxRate will not be used.

If an event update occurs which would otherwise send a notification when the maxRate has not expired, then a notification will be generated when the maxRate period has expired. Since EIDO conveys current state, if intermediate notifications are not sent then the subscriber will receive current state when the notification is sent after the period expires.

The maxRate is applicable to all subscription types. For a subscription of type “new”, when maxRate is specified and new notifications for new incidents matching the qualFilter (if supplied) exceed the maxRate, notifications are delayed until the maxRate period lapses. At that time, one notification is sent, containing one EIDO for every incident that passed the qualFilter (if specified) at any time during that period, and containing the current state of that incident. The incident state may be closed if the incident was opened and closed before the maxRate timer elapsed. In this case, maxRate delays the notifications and batches them but does not eliminate them.

For each subscription of type “single”, when the maxRate period lapses and the incident changed state, a notification is sent. The EIDO contains the current state of the incident at the time the notification is sent. This means that intermediate state changes are not sent and maxRate may reduce the number of EIDOs sent to the subscriber. An EIDO specifying
that an Incident is closed is not subject to the maxRate timer. Upon Incident close, the EIDO is immediately sent after which the subscription will be terminated.

### 2.2.2.1 Subscriber Behavior

A subscriber that wishes to request a specific maximum notification rate MUST populate the maxRate field in the subscribe message with a value greater than 0. The subscriber must be aware that the subscribe response message contains the actual maxRate to be used for the subscription.

### 2.2.2.2 Notifier Behavior

A notifier MUST consider the received maxRate parameter from the subscribe message and apply local policy to determine if the received maxRate is permitted. The notifier MAY decide to increase the proposed maxRate period based on its local policy, static configuration, or other implementation-determined constraints. The notifier MUST specify the effective maximum rate of notifications in the maxRate parameter of the subscribe response message.

A notifier MUST NOT generate a notification if the period since the most recent notification is less than the value of the maxRate parameter. Upon establish of a subscription or renewal thereof, a notification will be sent if required and the timer associated with the maxRate will be reset.
### 2.2.3 Subscription `maxRate` Example

<table>
<thead>
<tr>
<th>Subscriber</th>
<th>Notifier</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. Subscribe Request (maxRate: 30s, incidentId: E1)</td>
<td></td>
</tr>
<tr>
<td>2. Subscribe Response (maxRate: 30s, subscriptionId: 1000)</td>
<td></td>
</tr>
<tr>
<td>3. Event (incident: E1)</td>
<td></td>
</tr>
<tr>
<td>4. Event Response</td>
<td></td>
</tr>
<tr>
<td>7. Event (incident: E1)</td>
<td></td>
</tr>
<tr>
<td>8. Event Response</td>
<td></td>
</tr>
<tr>
<td>9. Re-subscribe Request (subscriptionId: 1000)</td>
<td></td>
</tr>
<tr>
<td>10. Subscribe Response (maxRate: 30s, subscriptionId: 1000)</td>
<td></td>
</tr>
<tr>
<td>11. Incident E1 Updated</td>
<td></td>
</tr>
<tr>
<td>12. Event (incident: E1)</td>
<td></td>
</tr>
<tr>
<td>13. Event Response</td>
<td></td>
</tr>
<tr>
<td>14. Unsubscribe (subscriptionId: 1000)</td>
<td></td>
</tr>
<tr>
<td>15. Unsubscribe Response (subscriptionId: 1000)</td>
<td></td>
</tr>
<tr>
<td>16. Subscribe Request (maxRate: 15s, incidentId: E1)</td>
<td></td>
</tr>
<tr>
<td>17. Subscribe Response (maxRate: 20s, subscriptionId: 1001)</td>
<td></td>
</tr>
<tr>
<td>18. Event (Events: E1)</td>
<td></td>
</tr>
<tr>
<td>19. Event Response</td>
<td></td>
</tr>
<tr>
<td>20. <code>maxRate</code> Expires</td>
<td></td>
</tr>
<tr>
<td>21. Incident E1 Updated</td>
<td></td>
</tr>
<tr>
<td>22. Event (incident: E1)</td>
<td></td>
</tr>
<tr>
<td>23. Event Response</td>
<td></td>
</tr>
</tbody>
</table>

© 2023 National Emergency Number Association, Inc.
1. The subscriber sends a subscribe request to the notifier requesting to receive updates about incident E1 at a maxRate interval of 30 seconds.

2. The notifier sends a subscribe response message back to the subscriber. The notifier sets the maxRate attribute to 30 seconds to confirm that a maxRate interval of 30 seconds will be used in the conversation. The subscription is given an ID of 1000.

3. The notifier immediately sends an event message to the subscriber with the latest incident information for incident E1.

4. The subscriber responds with an event response message indicating that it successfully received the event message.

5. Incident E1 is updated. An event message is not generated since the maxRate interval has not expired.

6. Incident E1 is updated. An event message is not generated since the maxRate interval has not expired.

7. The maxRate interval elapses. The notifier sends an event message to the subscriber with the most recent EIDO state for incident E1 (i.e., The contents of the EIDO object for incident E1 will be the incident state after the updates that occurred in Steps 5 and 6).

8. The subscriber responds with an event response message indicating that it successfully received the event message.

9. The subscriber sends a re-subscribe request.

10. The notifier sends a subscribe response message back to the subscriber. The maxRate stays the same as negotiated in the initial subscribe request. The notifier does not reset the maxRate timer. (Note: a re-subscribe request maintains the subscription beyond the agreed duration of the subscription)

11. Incident E1 is updated. An event message is not generated since the maxRate interval has not expired.

12. The maxRate interval elapses. The notifier sends an event message to the subscriber with the most recent EIDO state for incident E1.

13. The subscriber responds with an event response message indicating that it successfully received the event message.

14. The subscriber would like to change the interval associated to the maxRate timer. The subscriber sends an unsubscribe message to the notifier.

15. The notifier sends an unsubscribe response message back to the subscriber indicating that the subscription has been cancelled.

16. The subscriber sends a subscribe request to the notifier requesting to receive updates about incident E1 at a maxRate interval of 15 seconds.

17. The notifier sends a subscribe response message back to the subscriber. Because of local policies the notifier sets the maxRate attribute to 20 seconds. This is the maxRate interval that will be used during the conversation. The subscription ID is now 1001.
18. The notifier immediately sends an event message to the subscriber with the latest incident information for incident E1.
19. The subscriber responds with an event response message indicating that it successfully received the event message.
20. The maxRate timer expires. A notify message is not generated because incident E1 has not been updated.
21. Incident E1 is updated
22. Since the maxRate timer has already expired the update of incident E1 automatically generates a notify event that is sent to the subscriber.
23. The subscriber responds with an event response message indicating that it successfully received the event message.

2.3 qualFilter

A qualFilter MAY be included in the filter specification of the subscription. It specifies the criteria to be applied to an incident for a subscriber to be notified of that incident by being sent an EIDO for that incident. Example criteria include common priorities, geospatial boundaries, common incident types, and common incident statuses, as defined by the EIDO Standard.

The qualFilter is the primary mechanism to control which incidents are sent to the subscribed for a given subscription. The state of an incident can change such that an incident that did not originally pass the qualFilter may pass subsequently as the state of an incident changes. For example, the location of the incident may change which would trigger evaluation of the qualFilter. The notifier is NOT required to dereference location-by-reference that is mentioned in a qualFilter every time the filter is evaluated. If the notifier does not dereference the location, it SHALL use the last location it obtained for its own purposes when evaluating the filter.

The qualFilter is relatively expensive for a notifier to compute. The more complex the qualFilter, the more compute load is required for its evaluation. Rerunning the qualFilter for each subscriber for each change in the state of an incident may require more resources than a notifier has available for that purpose. To mitigate that, an evaluation rate MAY be negotiated between the subscriber and the notifier. The subscriber specifies the qualFilterEvalPeriod, which is the integer number of seconds between per-incident evaluations requested by the subscriber. The notifier's response to the subscription request SHALL specify the qualFilterEvalPeriod which MUST be greater than or equal to the value in the subscription request. The notifier attempts to rerun the qualFilter on an incident it is considering for an EIDO notification at the negotiated rate and sends one EIDO for an incident for which:
a) the qualFilter passes, and
b) it has not previously sent an EIDO to the subscriber for that incident for any reason.

The notifier MAY evaluate the filter more frequently than the negotiated period if the necessary resources are available.

If the notifier’s workload increases after the qualFilterEvalPeriod is negotiated, the notifier may not be able to keep up with the negotiated rate. It is RECOMMENDED that the notifier keep the maximum subscription duration in the range of 15-30 minutes, which would allow it to renegotiate a new qualFilterEvalPeriod reasonably promptly in such a circumstance. Subscribers should not expect the qualFilterEvalPeriod specified in the response to remain constant from subscription to subscription for this reason.

In addition, notifiers MUST implement a scheduling mechanism. Notifiers MUST track the qualFilter compute time for incidents for each subscription containing a qualFilter. A Policy (qualFilter) provides a weight factor as a real number with a default value of 1.0. The notifier SHALL have a provisioned value that specifies the percentage of the total CPU utilization that the cumulative qualFilter evaluations for an incident is allowed to use. Each subscription SHALL be allowed to consume a weighted fraction of the allowed time. If all weights were 1.0 and all negotiated qualFilterEvalPeriod values were the same or no subscriptions have a qualFilterEvalPeriod, all qualFilter instances would use the same compute time and simpler filters would always run more often than complex filters on a saturated system.

The weight factors allow management to effect an increase or decrease in the rate a qualFilter will be run. This can be accomplished by ordering the subscriptions by the average compute time for each subscription multiplied by its qualFilter weight factor with those subscriptions whose qualFilterEvalPeriod has not elapsed placed at the end ordered by the time of last evaluation and evaluating the incident for each subscription in that order until such time that the provisioned limit is exhausted. Any subscription not evaluated will have its compute time for that evaluation set to zero which cause the average to be lowered for that subscription. Other algorithms that achieve the same result are acceptable.

The notifier is NOT required to dereference location-by-value that is mentioned in a qualFilter every time the filter is evaluated. Rather, it uses the last location it obtained for its own purposes when evaluating the filter.

The recommended rate for qualFilterEvalPeriod is 60 seconds. Since the qualFilter may take substantial compute resource, if the notifier becomes busy after the qualFilterEvalPeriod is negotiated, it may not be able to keep up with the negotiated rate. It is RECOMMENDED that the notifier keep the maximum subscription duration in the range of 15-30 minutes, which would allow it to renegotiate the qualFilterEvalPeriod reasonably promptly in such a
circumstance. Subscribers should not expect the qualFilterEvalPeriod specified in the response to remain constant from subscription to subscription for this reason.

The qualFilter criteria SHALL be specified as Basic Regular Expressions as specified in POSIX Extended Regular Expressions as defined in section 9 of IEEE Std 1003.1-2017. Data elements in the EIDO SHALL be testable variables within the expression using JSONPath [24] notation. Geospatial boundaries are described with GML [17] shapes limited to those described in OGC 06-142r1 [5]. Coordinate references MUST be EPSG 4326.

A qualFilter is a m4 [22] macro expanding to a POSIX expression where the “^” (caret) character is used as a location/shape overlap operator.

When searching text elements, partial string matches and whole “word” matches MUST be possible.

Literal strings in text elements SHALL be compared in a case-insensitive manner. A literal string in the qualFilter MUST be upper case.

Note: The POSIX expr standard does not provide for case-insensitive comparisons.

As a common extension to the POSIX standard, whole word matches SHALL be accommodated in regular expressions by surrounding the search word with “\<” and “\>”, for example, “\<word\>”.

2.3.1 Applying a qualFilter

The following operations are performed on the filter value:

The first operation expands the filter value using the m4 processor. The effect of the m4 syscmd macro MUST be disabled to address the security risks associated with its use. This can be accomplished by setting the SHELL environment variable to /bin/false, performing a chroot and running the process as other than root, or equivalent. Given that m4 may recurse indefinitely, a limitation MUST be enforced to prevent such occurrence. Such a limitation may be effected by enforcing a time limit such as through the use of the POSIX timeout command or a memory limit.

The second operation replaces all instances of JSONPath references with a numeric, string, or location value obtained by locating the element in the EIDO pointed to by the JSONPath. Elements of an EIDO may contain references to other elements in the EIDO which must be dereferenced. A JSONPath reference is surrounded by a pair of bracket characters (“[” and “]”). In the case of an EIDO element that is a Boolean (i.e., “true” or “false”), the value will be zero or one. In the case of an EIDO element that is a string, the value shall be upper-cased. In the case of a geospatial reference (see below), the value will be an XML shape enclosed in braces (“{” and “}”). There is no ability to reference elements of XML documents other than PIDF-LOs [4][5][5] contained within JSON elements at this time.
The third operation compares each pair of geospatial references (see below) separated by the ^ (caret) character denoting the operator with 0 or 1 depending on whether the shape on the left side of the operator partially overlaps or contacts the shape on the right side.

The fourth operation submits the resulting value to the POSIX “expr” function. The expr exit value MUST be an integer. If the exit value is 0, the EIDO passes the filter. If it is non-zero, it fails.

If a geospatial reference is an EIDO element, then it must be a PIDF-LO. Otherwise, the geospatial reference must be literal GML polygon object or reference thereto. If the location object contains solely a civic address and does not contain geospatial coordinates, then the Geocode Service (GCS) as described in NENA-STA-010.3 SHALL be used to derive the geospatial coordinates that correspond to the civic address.

Geospatial references MUST use WGS84. The Spatial Reference System (SRS) URN must be specified as “urn:ogc:def:crs:EPSG::4326” which requires that the coordinate pairs be latitude followed by longitude.

Geospatial references are based on The OpenGIS® Geography Markup Language Encoding Standard (GML). The Geography Markup Language (GML) is an XML grammar for expressing geographical features.

2.3.2 Example
The following example reflects an implementation of the qualFilter whose requirements are specified in Appendix A.

define(`River_Location', `<?xml version="1.0" encoding="utf-8"?>
  <Polygon xmlns="http://www.opengis.net/gml"
    srsName="urn:ogc:def:crs:EPSG::4326">
    <exterior>
      <LinearRing>
        <posList>
          45.558679896560022 -122.30836276702803
          45.559079298970516 -122.32490527625748
          ...
          45.539768832065988 -122.30731697599759
          45.558679896560022 -122.30836276702803
        </posList>
      </LinearRing>
    </exterior>
  </Polygon>')

define(`I5_Bridge_Location', `<?xml version="1.0" encoding="utf-8"?>
  <Polygon xmlns="http://www.opengis.net/gml"
    srsName="urn:ogc:def:crs:EPSG::4326">
    <exterior>
      <LinearRing>
        <posList>
          ...
        </posList>
      </LinearRing>
    </exterior>
  </Polygon>')
define(`I205_Bridge_Location', `<?xml version="1.0" encoding="utf-8"?>
<Polygon xmlns="http://www.opengis.net/gml"
srsName="urn:ogc:def:crs:EPSG::4326">
<exterior>
<LinearRing>
<posList>
45.60377701030086 -122.55193944604855
45.599440140350225 -122.55117887140773
. . .
45.603550863181852 -122.5511218283097
45.60377701030086 -122.55193944604855
</posList>
</LinearRing>
</exterior>
</Polygon>')dnl
define(`River_Incident_Type',`.*\<\(WATER|RESCUE|MISBOATR|MCI|MAYDAY|FMARIN
|EXPLOSION|DROWN|DEATH|BRGCLPSE\)>.*')dnl
define(`Bridge_Incident_Type',`.*\<\(TSTOP|VEHABND|THAZ|PURSUIT|MCI|FVEH|EX
PLOSION|DISVEH|DEATH\)>.*')dnl
define(`River_Note_Keyword', `.*\<\(RIVER|DROWN|DROWNING|BOAT\)>.*')dnl
define(`Bridge_Note_Keyword', `.*\<\(CRASH|WRECK|ACCIDENT|FIRE\)>.*')dnl

genreduce(`(?<!\^)incidentComponent.locationReference.locationByValue\^\{River_Location\}\|"\|callComponent.locationReference.locationByValue\^\{River_Location\}\|"
emergencyResourceComponent.unitLocationReference.locationByValue\^\{River_Location\}
) &
genreduce(`\{incidentComponent.incidentTypeCommonRegistryText\} : River_Incident_Type
|`
\{notesComponent.notesActionComments\} : River_Note_Keyword |`
\{additionalDataComponent.notesReference.notesActionComments\} : River_Note_Keyword |
\{locationComponent.notesReference.notesActionComments\} : River_Note_Keyword
) &


( (incidentComponent.locationReference.locationByValue)^{I5_Bridge_Location}) | 
(callComponent.locationReference.locationByValue)^{I5_Bridge_Location} | 
(emergencyResourceComponent.unitLocationReference.locationByValue)^{I5_Bridge_Location} 
) 
& 
(incidentComponent.incidentTypeCommonRegistryText) : Bridge_Incident_Type 
| 
(notesComponent.notesActionComments) : Bridge_Note_Keyword | 
(additionalDataComponent.notesReference.notesActionComments) : Bridge_Note_Keyword | 
(locationComponent.notesReference.notesActionComments) : Bridge_Note_Keyword 
) 
) 
) 
) 
) 
( (incidentComponent.locationReference.locationByValue)^{I205_Bridge_Location}) | 
(callComponent.locationReference.locationByValue)^{I205_Bridge_Location} | 
(emergencyResourceComponent.unitLocationReference.locationByValue)^{I205_Bridge_Location} 
) 
& 
(incidentComponent.incidentTypeCommonRegistryText) : Bridge_Incident_Type 
| 
(notesComponent.notesActionComments) : Bridge_Note_Keyword | 
(additionalDataComponent.notesReference.notesActionComments) : Bridge_Note_Keyword | 
(locationComponent.notesReference.notesActionComments) : Bridge_Note_Keyword 
) 
) 
)

2.4 Transport in Call Signaling
An EIDO MAY be transported in the SIP call signaling, especially when transferring an emergency call as described in the section on bridging in NENA-STA-010.
2.5 EIDO Dereference Factory

The agency’s IDX MAY maintain an internal EIDO RESTful web service URL that, when an HTTP GET whose last element of the path is the incident ID is presented, returns a Dereference Service URI for the Incident [27]. Data Rights Management would control if the issuance of a URI from the Dereference Factory is permitted. The Dereference Factory MUST supply the same URI for a given requestor and incident so that the URI expiration mechanism cannot be circumvented.

For example:

https://psap.alllegany.pa.us/IncidentData/v1/dereference/urn:emergency:uid:incident:6955@psap.alleghany.pa.us

HTTP request:

GET
/IncidentData/v1/dereference/urn%3Aemergency%3Auid%3Aincident%3A6955%40psap.alllegany.pa.us HTTP/1.1
Host: psap.alleghany.pa.us

Result is:

https://psap.alleghany.pa.us/foobar/asdasdasdasdasd

HTTP request:

GET /foo/foobar/asdasdasdasdasd HTTP/1.1
Host: psap.alleghany.pa.us

Result is:

{I AM AN EIDO}

One of the following HTTP status codes and descriptions MUST be returned:

<table>
<thead>
<tr>
<th>Status Code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>200</td>
<td>OK</td>
</tr>
<tr>
<td>403</td>
<td>Forbidden</td>
</tr>
<tr>
<td>404</td>
<td>Not found</td>
</tr>
<tr>
<td>406</td>
<td>Not Acceptable</td>
</tr>
<tr>
<td>500</td>
<td>Internal Server Error</td>
</tr>
</tbody>
</table>
2.6 **EIDO Dereference Service**

Any entity in possession of the above-referenced EIDO URI may issue an HTTP GET on that URI and a full EIDO will be returned, although normal authentication and authorization mechanisms as described above MUST be used to limit what entities may do any form of dereference. Once used, the URI expires for the dereferencer and may not be used again by that dereferencer. The Dereference service MUST NOT provide more than one dereference for the URI from a given dereferencer. If updates to the incident are required, use the subscription mechanism.

One of the following HTTP status codes and descriptions MUST be returned:

<table>
<thead>
<tr>
<th>Status Code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>200</td>
<td>OK</td>
</tr>
<tr>
<td>403</td>
<td>Forbidden</td>
</tr>
<tr>
<td>404</td>
<td>Not found</td>
</tr>
<tr>
<td>406</td>
<td>Not Acceptable</td>
</tr>
<tr>
<td>500</td>
<td>Internal Server Error</td>
</tr>
</tbody>
</table>

2.6.1 **Request and Response**

The GET request MUST contain an Accept: header which specifies the MIME type assigned to EIDO (application/emergency.eido+json) and MUST include as a parameter a comma-delimited list of the major version(s) of the schema the client supports (for example, Accept: application/emergency.eido+json; version="1,2,3"). If the server can fulfil the request, the response MUST include one and only one EIDO instance in the body of the reply and its version MUST be the highest mutually compatible major version. The client MUST expect to receive an object derived from any minor version of the specified EIDO schema, including a higher minor version than it currently supports. The client MUST ignore any fields it does not understand. If the server does not support any of the major versions found in the Accept: header of the GET request, it MUST return a 406 Not Acceptable response.

2.7 **EIDO Retrieval Service**

An agency’s IDX MUST provide an EIDO Retrieval Service which is a RESTful web service URL that, when an HTTP GET whose last element of the path is the incident ID is presented, returns an EIDO. The content of the EIDO returned is limited by Data Rights
Management policy. The EIDO Retrieval Service URL for an Agency can be found in the Agency Locator record for that agency as specified in NENA-STA-0101.

The requirements defined in section 2.6.1 above also apply to this service.

For example:

https://idx.esinet.net/IncidentDate/v1/incidents/urn:emergency:uid:incident:6955@psap.alleghany.pa.us

HTTP request:

GET /IncidentData/V1/incidents/urn%3Aemergency%3Auid%3Aincident%3A6955%40psap.alleghany.pa.us HTTP/1.1
Host: psap.alleghany.pa.us

Result is:

{I AM AN EIDO}

One of the following HTTP status codes and descriptions MUST be returned:

<table>
<thead>
<tr>
<th>Status Code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>200</td>
<td>OK</td>
</tr>
<tr>
<td>403</td>
<td>Forbidden</td>
</tr>
<tr>
<td>404</td>
<td>Not found</td>
</tr>
<tr>
<td>406</td>
<td>Not Acceptable</td>
</tr>
<tr>
<td>500</td>
<td>Internal Server Error</td>
</tr>
</tbody>
</table>

2.8 Data Rights Management

All FEs in an agency MUST have the same Data Rights Management (DRM) policy and therefore it is not important to know which FE created any particular data component. The only FE that can include a data component in an EIDO that it is sending, which it received in an EIDO, is the IDX, which will be defined in future work. An agency IDX MUST NOT send data received from another agency, except as noted below. An inter-agency IDX MAY send data received from other agencies but MUST respect the data rights management policy of the FE (typically an IDX) that sent it the data.

The EIDO Incident component contains data vital to any agency participating in an incident, therefore, any agency MAY include in an EIDO it sends, information from the

---

1 Anticipated to be in NENA-STA-010.4.
Incident data component it received from another agency, subject to its own data rights management policy.

Similarly, subject to its own data rights management policy, an agency receiving an EIDO from another agency related to an incident in which three or more agencies are involved MUST include the Dispatch data component information it received that identifies all of the agencies involved.

The content of an EIDO, regardless of the way it is conveyed, MUST comply with the data rights management policy of the agency that created the data being incorporated into the EIDO, controlled by the DRM mechanism defined in NENA-STA-010. The data rights management policy MUST conform to local, state/provincial, and federal laws and regulations. If an entity receives an EIDO from another agency, and subsequently sends an EIDO to another agency, it MUST filter the EIDO it sends to comport with the EIDO policy of the agency from which it received the information with respect to the data components it is copying.

Filtering of EIDO content to conform with DRM policy SHALL be accomplished by deleting data components in their entirety from the EIDO being sent, based on the policy. Partial filtering of individual data elements within components in an EIDO may be defined in a future edition of this document.

In the DRM policy language as described in NENA-STA-010, policy rules are expressed in an “if [condition] then [action]” statement. The action “permit” allows a data component to be included in the EIDO, while “deny” disallows that component from being included in the EIDO. The name of the data component prefixed by “EIDO.” is used to name the resource to which the rule refers, for example, “EIDO.Agent”. A new entry in the Interface Names registry specified in NENA-STA-010 is listed in Section 3.2, EIDOConveyance, which is used to control the access rights for EIDO data components.

2.9 Logging

Regardless of transport mechanism (i.e., subscribe, message, dereference, or included in a call transfer):

Any FE that sends or receives an EIDO MUST log it to a Logger FE as defined in NENA-STA-010 with an EIDO LogEvent [26].

Certain LogEvents are defined in NENA-STA-010. One of those, EidoLogEvent, is extended. In addition, the following LogEvent types are defined, some of which have common members: The “direction” member has one of two values, “incoming” and “outgoing”. The “queryId” member is used to relate the query to the response. The value of the “queryId” member is generated locally, MUST be globally unique, and is suggested to be of the form “urn:nena:uid:logEvent:”, followed by a locally unique ID, followed by a colon, followed by
the domain of the entity performing the logging, for example, “urn:nena:uid:logEvent:a99dasdas37:psap.example.com”. In most of these events, a “peerId” member is included, which value is the identity of the peer. For a client or subscriber, it is the identity of the server or notifier. For the server or notifier, it is the identity of the requesting client or subscriber.

### 2.9.1 EidoLogEvent

The EidoLogEvent is extended to include a “peerId” member containing the identity of the peer entity, and a “subscriptionId” member. Implementations of this standard MUST populate the “peerId” member and MUST populate the “subscriptionId” member when the EidoLogEvent results from a subscription. The “peerId” member will be required in a future version of the schema.

### 2.9.2 EidoDereferenceFactoryQueryLogEvent

Client requests using the EIDODereferenceFactory method MUST be logged using a EidoDereferenceFactoryQueryLogEvent which MUST include the “peerId” member containing the identity of the peer entity, an “incidentId” member in the header field of the log record, the “direction” member, and the “queryId” member. The response to the request, logged via a EidoDereferenceFactoryQueryResponseLogEvent, MUST include the “direction” member, the “queryId” member, and either a “generatedURI” member or a “responseError” member that contains an error code from the Error Codes Registry in the case of an error.

### 2.9.3 EidoDereferenceFactoryQueryResponseLogEvent

The response to an EIDO Dereference Factory request MUST be logged using a EidoDereferenceFactoryQueryResponseLogEvent. The log event MUST include the “direction” member, the “queryId” member, the “peerId” member. The log event must include a “generatedURI” member or a “responseError” member that contains an error code from the Error Codes Registry in the case of an error.

### 2.9.4 EidoDeniedLogEvent

If a client has requested an EIDO from the Dereference Service and that request has been denied or deemed invalid, it MUST be logged using an EidoDeniedLogEvent by both the Dereference Service and the dereferencer and MUST include an “eidoDereference” member containing the identity of the peer entity, an “incidentId” member in the header field of the log record, a “reasonCode” member containing an associated reason code from the EIDO Dereference Denial registry and a “reasonText” member.
2.9.5 EidoTransmissionErrorLogEvent
If an error in the process of sending or receiving an EIDO occurs, the error MUST be logged using an EidoTransmissionErrorLogEvent which MUST include a “peerId” member containing the identity of the peer entity, a “direction” member, an “incidentId” member in the header field of the log record, and a “errorCode” member containing a status code from the EIDO Transmission Status registry indicating the resultant failure and an “errorText” member. Note that a successful transmission will be logged with an EidoLogEvent.

2.9.6 SubscriptionRequestedLogEvent
The subscriber MAY and the notifier MUST log a subscription requested log event. The SubscriptionRequestedLogEvent MUST include a “peerId” member containing the identity of the peer entity, a “direction” member and a “queryId member. The log event may include the requested expiry time in the “expires” member, the requested incident ID in the “incidentId” member in the header field of the log record, and the requested qualFilter in the “qualFilter” member.

2.9.7 SubscriptionRequestedResponseLogEvent
The subscriber MAY and the notifier MUST log a subscription requested response log event. The SubscriptionRequestedResponseLogEvent MUST include a “direction” member and a “queryId member”. If the subscribe request was successful, the log event MUST include the expiry time used for the duration of the subscription in the “expires” member and the subscription ID in the “subscriptionId” member. If the subscribe request was denied the log event MUST include an error message in the “errorCode” and “errorText” members respectively.

2.9.8 SubscriptionTerminatedLogEvent
When a subscription is terminated because the incident has closed or the subscription expires or the subscriber asks for the subscription to be terminated via unsubscribe, the subscriber MAY send a SubscriptionTerminatedLogEvent, and the notifier MUST send a SubscriptionTerminatedLogEvent. The SubscriptionTerminatedLogEvent MUST include a “peerId” member containing the identity of the peer entity, a “direction” member, a “subscriptionId” member in the header field of the log record, and a “reason” member describing the reason why the subscription was terminated. The list of reasons includes “incident closed”, “expired”, “unsubscribed”, and “internal error”. Other reasons MAY use non-standard values. If the subscription being terminated is a single incident subscription the “incidentId” member must be populated.
2.9.9 SubscriptionTerminatedResponseLogEvent
The subscriber MAY and the notifier MUST log a terminate response log event. The TerminateResponseLogEvent MUST include a "peerId" member, a "direction" member, a "queryId" member, a "subscriptionId" member, a "statusCode" member and a "statusText" member.

2.9.10 WebSocketEstablishedLogEvent
The client MAY and the server MUST log a WebSocketEstablishedLogEvent which MUST include a "subscriber" member containing the identity of the peer entity, a "direction" member, a "websocketId" member to relate the WebSocket establishment and its associated termination, and a "statusCode" member containing a "statusDescription" member containing an HTTP status code and description. The value of the "websocketId" member is generated locally, MUST be globally unique, and is suggested to be of the form “urn:nena:uid:logEvent:”, followed by a locally unique ID, followed by a colon, followed by the domain of the entity performing the logging, for example, “urn:nena:uid:logEvent:a99dasdas37:psap.example.com”.

2.9.11 WebSocketTerminatedLogEvent
The client MAY and the server MUST log a WebSocketTerminatedLogEvent which MUST include a "subscriber" member containing the identity of the peer entity, a "direction" member, a "websocketId" member with the same value as that specified in its associated WebSocketEstablishedLogEvent, a "closeCode" member containing a WebSocket Close frame status code, and an optional "closeText" member containing a reason per RFC 6455 [12].

2.10 Security
All transports specified in this document are protected by TLS using credentials (for those entities within an ESInet) traceable to the PSAP Credentialing Agency (PCA).

Data rights management as specified above limits the information that may be included in any given EIDO.

3 IANA Considerations

3.1 Event Package Registry
IANA is requested to add the following value to the Event Package Registry:

<table>
<thead>
<tr>
<th>Event Package</th>
<th>Description</th>
<th>Reference</th>
</tr>
</thead>
<tbody>
<tr>
<td>EIDOCOnveyance</td>
<td>EIDO Conveyance Event Package</td>
<td>&lt;this document&gt;</td>
</tr>
</tbody>
</table>
3.2 **Interface Names Registry**
IANA is requested to add the following values to the Interface Names Registry.

<table>
<thead>
<tr>
<th>Name</th>
<th>Reference</th>
</tr>
</thead>
<tbody>
<tr>
<td>EIDOConveyance</td>
<td>&lt;this document&gt;</td>
</tr>
</tbody>
</table>

3.3 **EIDO Dereference Denial Reason Registry**
IANA is requested to add the following values to the EIDO Dereference Denial Reason Registry.

<table>
<thead>
<tr>
<th>Reason</th>
<th>Description</th>
<th>Reference</th>
</tr>
</thead>
<tbody>
<tr>
<td>Unknown</td>
<td>Unknown incident or URL not valid</td>
<td>&lt;this document&gt;</td>
</tr>
<tr>
<td>Policy</td>
<td>Policy did not permit EIDO dereference</td>
<td>&lt;this document&gt;</td>
</tr>
<tr>
<td>Authentication</td>
<td>Authentication of the dereferencer failed</td>
<td>&lt;this document&gt;</td>
</tr>
<tr>
<td>Closed</td>
<td>Incident is closed</td>
<td>&lt;this document&gt;</td>
</tr>
<tr>
<td>Merged</td>
<td>Incident was merged</td>
<td>&lt;this document&gt;</td>
</tr>
<tr>
<td>Expired</td>
<td>The URI used for dereference has expired</td>
<td>&lt;this document&gt;</td>
</tr>
<tr>
<td>Other</td>
<td>Other reason not listed above</td>
<td>&lt;this document&gt;</td>
</tr>
</tbody>
</table>

3.4 **EIDO Transmission Failure Registry**
IANA is requested to add the following values to the EIDO Transmission Status Registry.

<table>
<thead>
<tr>
<th>Reason</th>
<th>Description</th>
<th>Reference</th>
</tr>
</thead>
<tbody>
<tr>
<td>Unknown</td>
<td>Unknown host</td>
<td>&lt;this document&gt;</td>
</tr>
<tr>
<td>Route</td>
<td>Routing failure</td>
<td>&lt;this document&gt;</td>
</tr>
<tr>
<td>Authentication</td>
<td>Authentication of the dereferencer failed</td>
<td>&lt;this document&gt;</td>
</tr>
<tr>
<td>Policy</td>
<td>Policy of recipient did not allow transmission</td>
<td>&lt;this document&gt;</td>
</tr>
<tr>
<td>Invalid</td>
<td>Recipient Claims URL invalid</td>
<td>&lt;this document&gt;</td>
</tr>
<tr>
<td>Other</td>
<td>Other reason not listed above</td>
<td>&lt;this document&gt;</td>
</tr>
</tbody>
</table>

3.5 **WebSocket Subprotocol Name Registry**
IANA is requested to add the following values to the WebSocket Subprotocol Name Registry.
4 Impacts, Considerations, Abbreviations, Terms, and Definitions

4.1 Operations Impacts Summary
Inter-agency agreements and standard operating procedures will need to be established or revised to address the operational impact of the exchange of EIDOs between agencies. Agencies will be required to analyze their needs with respect to the implementation of the publishing and subscribing of EIDOs.

4.2 Technical Impacts Summary
The exchange of EIDOs between functional elements introduces a significant increase in the complexity of interoperable sharing of data relating to incidents. Introducing standardized data consisting of standardized elements inside an agency or between agencies fundamentally affects design and interoperability of components. Subsystems and interfaces within agencies will need to accommodate these specifications.

Network connectivity must be established between parties. In some circumstances, large volumes of EIDOs may be exchanged and the network and logging capacity to handle the volume of data must be planned for. Monitoring of performance will be required.

4.3 Security Impacts Summary
Credentials need to be provisioned in agencies and functional elements exchanging EIDOs. Agencies will need to create data rights management policies to determine what can be conveyed in EIDOs.

Some entities sending or receiving EIDOs may be outside the ESInet, e.g., tow truck operators. Such entities may need special consideration to ensure appropriate security measures are in place, e.g., firewalls, etc.

4.4 Recommendation for Additional Development Work

4.4.1 Unsolicited EIDO Conveyance
This first release of the EIDO Conveyance Standard only defines how EIDOs are conveyed in a subscription use case. A future revision of this standard will add support for unsolicited use cases (i.e., when an EIDO needs to be “pushed” to another party). This includes defining the behavior of the responder when only some of the resources can be provided.
per responder’s policy and defining the conditions by which a recipient may respond with insufficient information.

4.4.2 Deltas
Consideration should be given to extending the standard to support EIDO deltas.

4.4.3 Location and Content Filters
Consideration should be given to incorporating location (derived from RFC 6447 [11]) and content (derived from RFC 4661 [9], exclusive of “changed”) filters when subscribing to specific incidents.

4.4.4 qualFilter Examples
Consideration should be given to adding one or more examples of a qualFilter that include an area polygon, as well as a value selector like the content filter from RFC 4661 [9].

4.4.5 Signatures
Consideration should be given to non-repudiation of sending and receiving via signing and the negotiation of the requirement thereof.

4.4.6 Discrepancy Reporting
Consideration should be given to discrepancy reporting related to the structure or content of the EIDO or the conveyance mechanisms defined herein.

4.4.7 Subscribe to Resource
Consideration should be given to subscribing to emergency resources that are not assigned to an incident, in which case the incident number would be 0.

4.4.8 Removal of EIDO Dereference Factory and EIDO Dereference Service
Consideration should be given to the removal of the EIDO Dereference Factory and the EIDO Dereference Service as they duplicate functionality achieved via the subscription and notification mechanisms described herein.

4.5 Anticipated Timeline
Implementation and deployment will take place as required.

4.6 Cost Factors
This standard should result in a long-term substantial reduction in the cost of new interfaces that exchange incident data between functional elements of multiple vendors.
4.7 **Cost Recovery Considerations**
Not applicable.

4.8 **Additional Impacts (non-cost related)**
None known at this time.

4.9 **Abbreviations, Terms, and Definitions**
See NENA Master Glossary of 9-1-1 Terminology, NENA-ADM-000 *Error! Reference source not found.*, for a complete listing of terms used in NENA documents. All abbreviations used in this document are listed below, along with any new or updated terms and definitions.

<table>
<thead>
<tr>
<th>Term or Abbreviation (Expansion)</th>
<th>Definition / Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>EDXL-RM (Emergency Data Exchange Language – Resource Messaging)</td>
<td>An OASIS standard that describes a suite of standard messages for sharing data among information systems that coordinate requests for emergency equipment, supplies, and people.</td>
</tr>
<tr>
<td>EIDO (Emergency Incident Data Object)</td>
<td>A JSON-based (JavaScript Object Notation) object that is used to share emergency incident information between and among authorized entities and systems. NENA has adopted the JSON based EIDO (Emergency Incident Data Object) for sharing incident information among authorized NG9-1-1 entities and systems.</td>
</tr>
<tr>
<td>Expr</td>
<td>expr is a command line utility on Unix and Unix-like operating systems which evaluates an expression and outputs the corresponding value [21]</td>
</tr>
<tr>
<td>IDX (Incident Data eXchange)</td>
<td>A Functional Element that facilitates the exchange of Emergency Incident Data Objects (EIDOs) among other Functional Elements both within and external to an agency. (Previously called “IDE”.)</td>
</tr>
<tr>
<td>IEEE (Institute of Electrical and Electronics Engineers)</td>
<td>A professional association that develops, defines, and reviews electronics and computer science standards.</td>
</tr>
<tr>
<td>IETF (Internet Engineering Task Force)</td>
<td>Lead standard-setting authority for Internet protocols.</td>
</tr>
<tr>
<td>JSON (JavaScript Object Notation)</td>
<td>A lightweight data-interchange format based on a subset of the JavaScript Programming Language Standard ECMA-262</td>
</tr>
<tr>
<td>JSONPath</td>
<td>XPath for JSON</td>
</tr>
<tr>
<td>Term or Abbreviation (Expansion)</td>
<td>Definition / Description</td>
</tr>
<tr>
<td>---------------------------------</td>
<td>--------------------------</td>
</tr>
<tr>
<td>m4</td>
<td>m4 is a general-purpose macro processor included in most Unix-like operating systems and is a component of the POSIX standard [22]</td>
</tr>
<tr>
<td>Notifier</td>
<td>An element in an event notification mechanism that transmits events</td>
</tr>
<tr>
<td>OASIS (Organization for the Advancement of Structured Information Standards)</td>
<td>A standards development organization that promulgates standards for data interchange.</td>
</tr>
<tr>
<td>OGC (Open Geospatial Consortium)</td>
<td>A standards development organization that promulgates standards for the global geospatial community.</td>
</tr>
<tr>
<td>POSIX (Portable Operating System Interface)</td>
<td>A set of formal descriptions that provide a standard for the design of operating systems, especially ones which are compatible with UNIX. IEEE 1003, ISO/IEC 9945</td>
</tr>
<tr>
<td>Push</td>
<td>The process of sending an object, for example, an EIDO, from one functional element to another on an unsolicited basis</td>
</tr>
<tr>
<td>regex</td>
<td>A Regular Expressions (RE) provides a mechanism to select specific strings from a set of character strings [23]</td>
</tr>
<tr>
<td>SIP (Session Initiation Protocol)</td>
<td>A protocol specified by the IETF (RFC 3261) that defines a method for establishing multimedia sessions over the Internet. Used as the call signaling protocol in VoIP, NENA i2 and NENA i3.</td>
</tr>
<tr>
<td>Subscriber</td>
<td>An entity that establishes a subscription to a Notifier to be notified of information related to that subscription.</td>
</tr>
<tr>
<td>WebSocket</td>
<td>The WebSocket Protocol enables two-way communication between a client running untrusted code in a controlled environment to a remote host that has opted-in to communications from that code.</td>
</tr>
<tr>
<td>XPath</td>
<td>XML Path Language is a query language for selecting nodes from an XML document.</td>
</tr>
</tbody>
</table>
5 References


[24] Internet Engineering Task Force. *JavaScript Object Notation (JSON) Path (work in progress)*


6 Exhibit

Not Applicable.
7 Appendix A: Subscription Filter Example

Multnomah County, Oregon is across the Columbia River from Clark County, Washington. In addition to the part of the Columbia River they share, there are two Interstate bridges across the river between the two counties, the I5 bridge and the I205 bridge.

In this example, the Clark County PSAP wants to receive some EIDOs from Multnomah County. Since they don’t want to receive all of Multnomah County’s EIDOs, they will configure a filter that Multnomah County will use to test each EIDO to see whether or not to send it to Clark County.

The filter will specify that the only EIDOs that Clark County wants to receive are those that are related to Columbia River emergency incidents or incidents on either of the Interstate bridges that may affect Clark County.

To implement this filter, an expression is needed to compare the EIDO element values with the filter to determine if there is a match. Any EIDO that matches the filter will be sent to the subscriber.

If this logical expression is “true” when the filter is compared to an EIDO, it is considered a match and the EIDO would be sent:

\[
\text{(EIDO River Location)} \land ((\text{River Incident Type}) \lor (\text{River Note Keyword})) \\
\lor \\
\text{(EIDO I5 Bridge Location)} \land ((\text{Bridge Incident Type}) \lor (\text{Bridge Note Keyword})) \\
\lor \\
\text{(EIDO I205 Bridge Location)} \land ((\text{Bridge Incident Type}) \lor (\text{Bridge Note Keyword}))
\]

Where:

**EIDO River Location** is true when:

Any of these elements in the EIDO are within the PIDF of the area encompassing the Columbia River for the area of the river shared by both counties:

- Incident Location \( \text{EIDO:Incident/EIDO:Location} \)
- Caller Location \( \text{EIDO:Call/EIDO:Location} \)
- Unit Location \( \text{EIDO:EmergencyResource/EIDO:UnitLocation} \)
River Incident Type is true when the Incident Type Common EIDO element (EIDO:Incident/EIDO:IncidentTypeCommonRegistryText) contains at least one of the following values:

- WATER (Water Rescue)
- RESCUE (Rescue)
- MISBOATR (Missing Boater)
- MCI (Mass Casualty Incident)
- MAYDAY (Mayday)
- FMARINE (Fire Marine)
- EXPLOSION (Explosion)
- DROWN (Drowning)
- DEATH (Death)
- BRGCLPSE (Bridge Collapse)

River Note Keyword is true if any of these EIDO notes fields:

- EIDO Notes (Header)  EIDO:Notes/EIDO:NotesActionComments
- Additional Data Notes  EIDO:AdditionalData/EIDO:Notes
- Incident Location Notes  EIDO:Location/EIDO:Notes

Include any of these keywords:

- River
- Drown
- Drowning
- Boat

EIDO I5 Bridge Location is true when:

Any of these elements in the EIDO are within the PIDF of the area encompassing the I5 bridge:

- Incident Location  EIDO:Incident/EIDO:Location
- Caller Location  EIDO:Call/EIDO:Location
- Unit Location  EIDO:EmergencyResource/EIDO:UnitLocation

Bridge Incident Type is true when the Incident Type Common EIDO element (EIDO:Incident/EIDO:IncidentTypeCommonRegistryText) contains at least one of the following values:

- TSTOP (Traffic Stop)
- VEHABND (Abandoned Vehicle)
THAZ (Traffic Hazard)
PURSUIT (Pursuit)
MCI (Mass Casualty Incident)
FVEH (Fire Vehicle)
EXPLOSION (Explosion)
DISVEH (Disabled Vehicle)
DEATH (Death)

**Bridge Note Keyword** is true if any of these EIDO notes fields:

<table>
<thead>
<tr>
<th>Field Type</th>
<th>EIDO Notes Field</th>
</tr>
</thead>
<tbody>
<tr>
<td>EIDO Notes (Header)</td>
<td>EIDO:Notes/EIDO:NotesActionComments</td>
</tr>
<tr>
<td>Additional Data Notes</td>
<td>EIDO:AdditionalData/EIDO:Notes</td>
</tr>
<tr>
<td>Incident Location Notes</td>
<td>EIDO:Location/EIDO:Notes</td>
</tr>
</tbody>
</table>

Include any of these keywords:

- Crash
- Wreck
- Accident
- Fire

**EIDO I205 Bridge Location** is true when:

Any of these location elements in the EIDO are within the polygon of the area encompassing the I205 Bridge:

<table>
<thead>
<tr>
<th>Field Type</th>
<th>EIDO Field</th>
</tr>
</thead>
<tbody>
<tr>
<td>Incident Location</td>
<td>EIDO:Incident/EIDO:Location</td>
</tr>
<tr>
<td>Caller Location</td>
<td>EIDO:Call/EIDO:Location</td>
</tr>
<tr>
<td>Unit Location</td>
<td>EIDO:EmergencyResource/EIDO:UnitLocation</td>
</tr>
</tbody>
</table>
I205 Bridge GML

<?xml version="1.0" encoding="utf-8"?>
<Polygon xmlns="http://www.opengis.net/gml"
srsName="urn:ogc:def:crs:EPSG::4326">
  <exterior>
    <LinearRing>
      <posList>
        45.60377701030086 -122.55193944604855
        45.59940140350225 -122.55117887140773
        45.59679318011158 -122.55041829586764
        45.59395382934661 -122.5489281868234
        45.59336704156807 -122.5475837686755
        45.587518673348143 -122.5459357237045
        45.58548272244974 -122.5449989626815
        45.584098766544912 -122.5418158021571
        45.582461928475254 -122.5439914365553
        45.58099804892089 -122.54397242218948
        45.579893460215089 -122.5412453711768
        45.578615836551648 -122.5442876697397
        45.577551128281982 -122.54479003992839
        45.575767695322952 -122.54572174476266
        45.574370190232173 -122.54669147832902
        45.5738455891029 -122.54729993804165
        45.573478430585169 -122.5465773912336
        45.574663004094418 -122.5456837160306
        45.576127048224748 -122.54479003992839
        45.577617672717452 -122.54412453711768
        45.579121566706988 -122.54364917706783
        45.580332636941705 -122.54342100467557
        45.581610221934227 -122.54334494721149
        45.582874469980879 -122.54336396157754
        45.583979000230727 -122.54353590987173
        45.585349651565195 -122.54389636472541
        45.586720269624777 -122.544466795706
        45.588077548040474 -122.54509426978467
        45.589341650396875 -122.54583583095882
        45.591217791767747 -122.54705275128339
        45.593373281451818 -122.54834572907208
        45.594637264198411 -122.54889714568668
        45.59610078852063 -122.54953888121435
        45.597431231166411 -122.54996195108316
        45.598409086510173 -122.55028519530549
        45.599786038494926 -122.5505983279446
        45.602114139555965 -122.55091267028348
        45.603550863181852 -122.5511218230397
        45.60377701030086 -122.55193944604855
      </posList>
    </LinearRing>
  </exterior>
</Polygon>
I5 Bridge I5 GML

<?xml version="1.0" encoding="utf-8"?>
<Polygon xmlns="http://www.opengis.net/gml"
srsName="urn:ogc:def:crs:EPSG::4326">
  <exterior>
    <LinearRing>
      <posList>
        45.621797702801814 -122.67273541470757
        45.621947310420069 -122.67318700567574
        45.620301602337122 -122.67422091216531
        45.616611049960056 -122.67641944969779
        45.614466428582887 -122.67771480439467
        45.6084706583942 -122.6811644814556
        45.607113873706339 -122.68191317245493
        45.605941209019647 -122.68245270542798
        45.605795299413614 -122.68190841953788
        45.6062043434555715 -122.68172302879469
        45.606468723452906 -122.68158042149923
        45.607396539618662 -122.68110268544052
        45.608414126111086 -122.68055007992223
        45.608867212651774 -122.68029754489572
        45.610517413747687 -122.67938247841863
        45.611714505819862 -122.67864567106085
        45.621797702801814 -122.67273541470757
      </posList>
    </LinearRing>
  </exterior>
</Polygon>
Columbia River GML

```xml
<?xml version="1.0" encoding="utf-8"?>
<Polygon xmlns="http://www.opengis.net/gml"
srsName="urn:ogc:def:crs:EPSG::4326">
<exterior>
<LinearRing>
<posList>
45.558679896560022 -122.30836276702803
45.559079298970516 -122.324905276253748
45.559611830721451 -122.33156030796181
45.564537511996662 -122.33954634708624
45.571991776599248 -122.34791267353103
45.575452348047463 -122.35513813711532
45.576178184826 -122.35989173221765
45.577315645300871 -122.37700467242763
45.577315645300871 -122.3899344941525
45.576650189254906 -122.39506833183799
45.57971122159563 -122.40400508926342
45.580775889395852 -122.40685724596511
45.58011047819389 -122.4152235724099
45.5755854432137 -122.43423795102058
45.57704963961657 -122.44697758434802
45.58822798323532 -122.510254636647
45.59408253268276 -122.5355845866593
45.603927418315777 -122.57304291200074
45.61190849585176 -122.61335339314456
45.61274505449104 -122.63179733987528
45.61642471922607 -122.64415668588231
45.615632601063737 -122.65632588822911
45.618026542391078 -122.66031890779135
45.6224151683254 -122.67762199166157
45.633318752958132 -122.70138996447525
45.63903563883684 -122.71222816030132
45.64328968669732 -122.72287621156789
45.6475434115992 -122.74109059017857
45.6500689075541 -122.74911605376286
45.656714407344045 -122.75615137368692
45.671198861238508 -122.76505813111235
45.677044735126458 -122.76851071969395
45.681428740344131 -122.76927129433471
45.691258289838231 -122.76775014415381
45.707991117196343 -122.76242611897015
45.719940070178154 -122.75919367404879
45.727108216371164 -122.75653166100733
45.735602605376016 -122.7563415713471
45.741176346211489 -122.75672180466751
45.747142968333389 -122.7586232430682
45.7598841207556 -122.7584330940797
45.76784297427389 -122.75995424958893
45.772352487977 -122.76185568709025
</posList>
</LinearRing>
</exterior>
</Polygon>
```
45.580809159814919 -122.55819268224644
45.577548552623682 -122.55248836884311
45.574487402471675 -122.546784055543979
45.57202505241429 -122.546784055543979
45.56716658467019 -122.52853025236919
45.565569118167558 -122.52263579530569
45.565302951070407 -122.51731176922272
45.563772063030342 -122.51056166478995
45.562440821786538 -122.50409677585448
45.561242678406927 -122.4957304494007
45.560310771225147 -122.48536761362442
45.559378848754932 -122.4800435875415
45.55912581980116 -122.47928301290068
45.55912581980116 -122.46968075176522
45.558313775360489 -122.46597294769333
45.558247207542649 -122.4631207909917
45.557980935371916 -122.46017356245989
45.557951198079333 -122.45085651681478
45.561042984845926 -122.44477191609104
45.562440821786538 -122.44249019036994
45.562440821786538 -122.43593023049573
45.562973322061225 -122.4242363872994
45.564903597276694 -122.41615527724423
45.567432812774143 -122.41615527724423
45.5678293796222 -122.3875338845811
45.569177131519439 -122.37733742428264
45.56423521682006 -122.36535836559608
45.560341279826275 -122.3593688367024
45.55145107421193 -122.34988541557942
45.548807655349094 -122.34104372908479
45.547409479364035 -122.33562463207107
45.539569062961959 -122.32179167134854
45.53916522055829 -122.31779865178629
45.53968947064285 -122.31273607391063
45.539768832065988 -122.30731697599759
45.558679896560022 -122.30836276702803
</posList>
</LinearRing>
</exterior>
</Polygon>
ACKNOWLEDGMENTS

The National Emergency Number Association (NENA) 9-1-1 Core Services Committee, EIDO Conveyance Working Group developed this document.

NENA recognizes the following industry experts and their employers for their contributions to the development of this document.

<table>
<thead>
<tr>
<th>Members</th>
<th>Employer</th>
</tr>
</thead>
<tbody>
<tr>
<td>Steve O’Conor, ENP, 9-1-1 Core Services Committee Co-Chair</td>
<td>Next Generation 9-1-1 Consulting Services, LLC</td>
</tr>
<tr>
<td>Terry Reese, 9-1-1 Core Services Committee Co-Chair</td>
<td>Ericsson, Inc.</td>
</tr>
<tr>
<td>Henry Unger, Working Group Co-Chair</td>
<td>Pulsiam</td>
</tr>
<tr>
<td>Tony Dunsworth, Working Group Co-Chair</td>
<td>City of Alexandria VA</td>
</tr>
<tr>
<td>Rick Blackwell, ENP, Working Group Co-Chair</td>
<td>Greenville County SC</td>
</tr>
<tr>
<td>Mary Borst</td>
<td>Mayo Clinic Medical Transport - Rochester MN</td>
</tr>
<tr>
<td>Susan Carr, ENP</td>
<td>City of Plano TX</td>
</tr>
<tr>
<td>Francois Cregheur</td>
<td>Motorola Solutions, Inc.</td>
</tr>
<tr>
<td>Nathan Daniels</td>
<td>Northrop Grumman</td>
</tr>
<tr>
<td>Jerry Eisner</td>
<td>RedSky Technologies</td>
</tr>
<tr>
<td>Mark Hamilton</td>
<td>Motorola Solutions, Inc.</td>
</tr>
<tr>
<td>Diane Harris, ENP</td>
<td>Zetron Inc.</td>
</tr>
<tr>
<td>Glenna Johnson</td>
<td>DeKalb County IL</td>
</tr>
<tr>
<td>Becky Jones</td>
<td>Bell Canada</td>
</tr>
<tr>
<td>Ryan Keeling</td>
<td>Bell Canada</td>
</tr>
<tr>
<td>James Kinney</td>
<td>INdigital</td>
</tr>
<tr>
<td>Julie Lobb</td>
<td>Carbyne, Inc.</td>
</tr>
<tr>
<td>Roger Marshall</td>
<td>Comtech Telecommunications Corporation</td>
</tr>
<tr>
<td>Steven McDowall</td>
<td>Carbyne, Inc.</td>
</tr>
<tr>
<td>Rich McQuade</td>
<td>Tyler Technologies</td>
</tr>
<tr>
<td>Greg Middleton</td>
<td>CentralSquare Technologies</td>
</tr>
<tr>
<td>Dan Mongrain</td>
<td>Motorola Solutions, Inc.</td>
</tr>
<tr>
<td>Katy Myers, ENP</td>
<td>Norcomm Public Safety Communication Inc.</td>
</tr>
<tr>
<td>Etienne Paquet</td>
<td>Komutel</td>
</tr>
<tr>
<td>Michael Riddle</td>
<td>Northrop Grumman</td>
</tr>
<tr>
<td>Brian Rosen</td>
<td>Individual-Brian Rosen</td>
</tr>
<tr>
<td>Driton Salihu</td>
<td>City of Montreal QC CA</td>
</tr>
<tr>
<td>Jerry Schlesinger, PMP</td>
<td>Intelligent Systems Inc.</td>
</tr>
</tbody>
</table>
Members | Employer
---|---
Matthew Schreiner | Mission Critical Partners Inc.
Bill Stevens | Zetron Inc
Brodie Taillefer | Versaterm Inc.
Matt Tenold | City of Lynnwood WA
Ravi Valavandan, ENP | City of Calgary AB CA
Michael Vislocky | Network Orange Inc.
Jason Wellonen | Carbyne, Inc.

Special Acknowledgements:

Delaine Arnold, ENP, Committee Resource Manager, has facilitated the production of this document through the prescribed approval process.

The EIDO Conveyance Standard Working Group is part of the NENA Development Group that is led by:

- Jim Shepard, ENP, and Wendi Rooney, ENP, Development Steering Council Co-Chairs
- Brandon Abley, ENP, Technical Issues Director
- April Heinze, ENP, 9-1-1 and PSAP Operations Director